Claims

What is claimed is:

1. A method for tracing an instrumented program, comprising:
   triggering an trap instruction in the instrumented program;
   transferring control of the instrumented program to a trap handler associated with the trap instruction; and
   emulating an instruction corresponding to the trap instruction in the trap handler, wherein the instruction relates to creating or dismantling a stack frame.

2. The method of claim 1, further comprising:
   replacing the instruction with the trap instruction in the instrumented program.

3. The method of claim 1, further comprising:
   calling into a tracing framework to execute an action associated with the trap instruction.

4. The method of claim 1, wherein emulating the instruction comprises emulating a pushl instruction.

5. The method of claim 4, wherein emulating a pushl instruction comprises:
   obtaining a stack pointer location, wherein the stack pointer location corresponds to a location in the stack frame;
   incrementing an instruction pointer to obtain an incremented instruction pointer;
   loading the incremented instruction pointer in the stack frame at one location before the stack pointer location;
   loading a CS value stored one location after the stack pointer location into the stack pointer location;
   loading an EFLAGS value stored two locations after the stack pointer location into one location after the stack pointer; and

loading a base pointer into two locations after the stack pointer location.

6. The method of claim 5, further comprising:
   decrementing the stack pointer location by one location; and
   issuing a return from interrupt instruction.

7. The method of claim 5, wherein the instruction pointer is loaded at the stack pointer location.

8. The method of claim 1, wherein emulating the instruction comprises emulating a enter instruction.

9. The method of claim 8, wherein emulating an enter instruction comprises:
   obtaining a stack pointer location, wherein the stack pointer location is corresponds to a location in the stack frame;
   incrementing an instruction pointer to obtain an incremented instruction pointer;
   loading the incremented instruction pointer in the stack frame at one location before the stack pointer location;
   loading a CS value stored one location after the stack pointer location into the stack pointer location;
   loading an EFLAGS value stored two locations after the stack pointer location into one location after the stack pointer;
   loading a base pointer into two locations after the stack pointer location; and
   loading the base pointer into a base pointer register.

10. The method of claim 9, further comprising:
    decrementing the stack pointer location by one location; and
    issuing a return from interrupt instruction.

11. The method of claim 9, wherein the instruction pointer is loaded at the stack pointer location.

12. The method of claim 1, wherein emulating the instruction comprises emulating a popl instruction.

13. The method of claim 12 wherein emulating a popl instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a location in the stack frame;

loading a base pointer obtained from three locations after the stack pointer location into a base pointer register;

loading an EFLAGS value stored two locations after the stack pointer location into three locations after the stack pointer location;

loading a CS value stored one location after the stack pointer location into two locations after the stack pointer location;

incrementing an instruction pointer to obtain an incremented instruction pointer; and

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location.

14. The method of claim 13, further comprising:

incrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

15. The method of claim 13, wherein the instruction pointer is loaded at the stack pointer location.

16. The method of claim 1, wherein emulating the instruction comprises emulating a leave instruction.

17. The method of claim 16 wherein emulating a leave instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a first location in the stack frame;

obtaining a base pointer location, wherein the base pointer location corresponds to a second location in the stack frame;

loading a base pointer obtained at the base pointer location into a base pointer register;

loading an EFLAGS value stored two locations after the stack pointer location into the base pointer location;

loading a CS value stored one location after the stack pointer location into one location before the base pointer location;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at two locations before the base pointer location; and

loading the instruction pointer at three locations before the base pointer.

18. The method of claim 17, further comprising:

setting the stack pointer location to three locations before the base pointer location;

incrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

19. A system for tracing an instrumented program, comprising:

the instrumented program comprising at least one trap instruction associated with an instruction, wherein the instruction relates to creating or dismantling a stack frame;

a thread configured to execute the instrumented program; and

a trap handler configured to halt execution of the thread when the trap instruction is encountered and to emulate the instruction associated with the trap instruction.

20. The system of claim 19, further comprising:

a tracing framework operatively connected to the trap handler configured to perform an action associated with the trap instruction.

21. The system of claim 19, wherein the instruction is replaced with a the trap instruction in the instrumented program.

22. The system of claim 19, wherein emulating the instruction comprises emulating a pushl instruction.

23. The system of claim 22, wherein emulating a pushl instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location is corresponds to a location in the stack frame;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location;

loading a CS value stored one location after the stack pointer location into the stack pointer location;

loading an EFLAGS value stored two locations after the stack pointer location into one location after the stack pointer; and

loading a base pointer into two locations after the stack pointer location.

24. The system of claim 23, further comprising:

decrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

25. The system of claim 23, wherein the instruction pointer is loaded at the stack pointer location.

26. The system of claim 19, wherein emulating the instruction comprises emulating a enter instruction.

27. The system of claim 26, wherein emulating an enter instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location is corresponds to a location in the stack frame;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location;

loading a CS value stored one location after the stack pointer location into the stack pointer location;

loading an EFLAGS value stored two locations after the stack pointer location into one location after the stack pointer;

loading a base pointer into two locations after the stack pointer location; and

loading the base pointer into a base pointer register.

28. The system of claim 27, further comprising:

decrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

29. The system of claim 27, wherein the instruction pointer is loaded at the stack pointer location.

30. The system of claim 19, wherein emulating the instruction comprises emulating a popl instruction.

31. The system of claim 30, wherein emulating a popl instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a location in the stack frame;

loading a base pointer obtained from three locations after the stack pointer location into a base pointer register;

loading an EFLAGS value stored two locations after the stack pointer location into three locations after the stack pointer location;

loading a CS value stored one location after the stack pointer location into two locations after the stack pointer location;

incrementing an instruction pointer to obtain an incremented instruction pointer; and

loading the incremented instruction pointer in the stack frame at one location before the stack pointer location.

32. The system of claim 31, further comprising:

incrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.

33. The system of claim 31, wherein the instruction pointer is loaded at the stack pointer location.

34. The system of claim 19, wherein emulating the instruction comprises emulating a leave instruction.

35. The system of claim 34, wherein emulating a leave instruction comprises:

obtaining a stack pointer location, wherein the stack pointer location corresponds to a first location in the stack frame;

obtaining a base pointer location, wherein the base pointer location corresponds to a second location in the stack frame;

loading a base pointer obtained at the base pointer location into a base pointer register;

loading an EFLAGS value stored two locations after the stack pointer location into the base pointer location;

loading a CS value stored one location after the stack pointer location into one location before the base pointer location;

incrementing an instruction pointer to obtain an incremented instruction pointer;

loading the incremented instruction pointer in the stack frame at two locations before the base pointer location; and

loading the instruction pointer at three locations before the base pointer.

36. The system of claim 35, further comprising:

setting the stack pointer location to three locations before the base pointer location;

incrementing the stack pointer location by one location; and

issuing a return from interrupt instruction.